XText based query engine for MongoEMF

Author: Tiger Gui

Mentor: Bryan Hunt

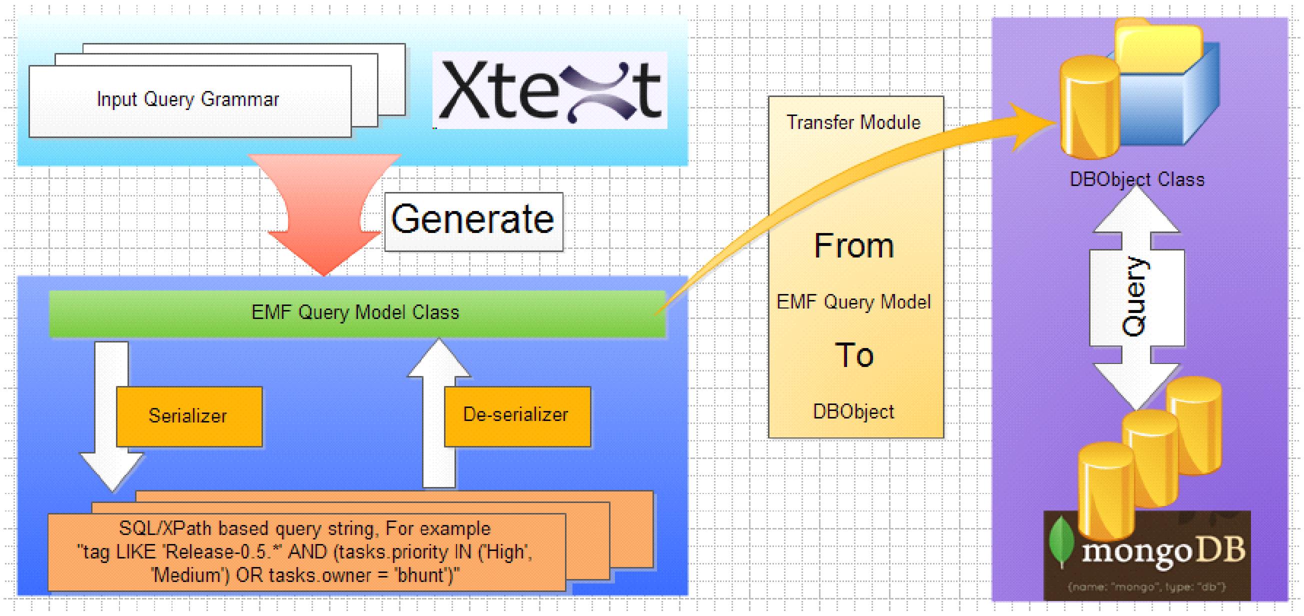
Abstract

Ed Merks and Bryan Hunt have created a project called MongoEMF that allows you to persist EMF objects to MongoDB. MongoEMF has the ability to query objects based on attribute and reference values, but is lacking some functionality and robustness. This project will re-write MongoEMF's query engine using Xtext and EMF, we will bring in some features in EMF Query 2 project, and try to contribute back in the future.

Detailed Information

The project will consist of two major parts. The first will be to define a query grammar using Xtext. The result will be an EMF query model that can be serialized and de-serialized to a string. The string will be used as the query portion of a URI, and must be human readable. For example: mongo://host/db/collection/?(tag == 'java' || tag == 'JSON') && (category == 'Eclipse'). Clients will be able to specify the query as string or EMF model. The second part of the project will be to create a processing engine that builds a MongoDB query from the EMF query model. The result will be a DBObject that can be sent to MongoDB as a query.

This is the project structure diagram.



As shown in the figure, the first part of job is building the XText "Input Query Grammar", the second part of job is building "Transfer Module".

Project Design & Explanation

1.XText Input Query Grammar Design

This new query engine for MongoEMF is designed similar with EMF Query 2:

EMF Query 2 use XText generated SQL like query string to query data from EMF XMI file, and our project will use XText generated SQL like query string to query MongoDB. We both use XText to generated SQL like query string, so this project can re-use some part of EMF Query 2's Xtext grammar. Of course, we should extending that to include MongoDB specific query features.

Bryan discussed with me about reaching out to the EMF Query 2 developers to get their feedback and discuss the possibility of contributing any non MongoDB specific enhancements back to EMF Query 2 project. Then we will communicate with EMF Query 2 team in the future, expect that we can contribute back something.

1.1 Human Readable Query String

Queries in MongoDB are expressed as JSON (BSON). Usually we think of query object as the equivalent of a SQL "WHERE" clause Mongo Query Language:

```
db.users.find({ x:3, y:"abc"}).sort({x:1});// select * from users where x=3 and y='abc' order by x asc;
```

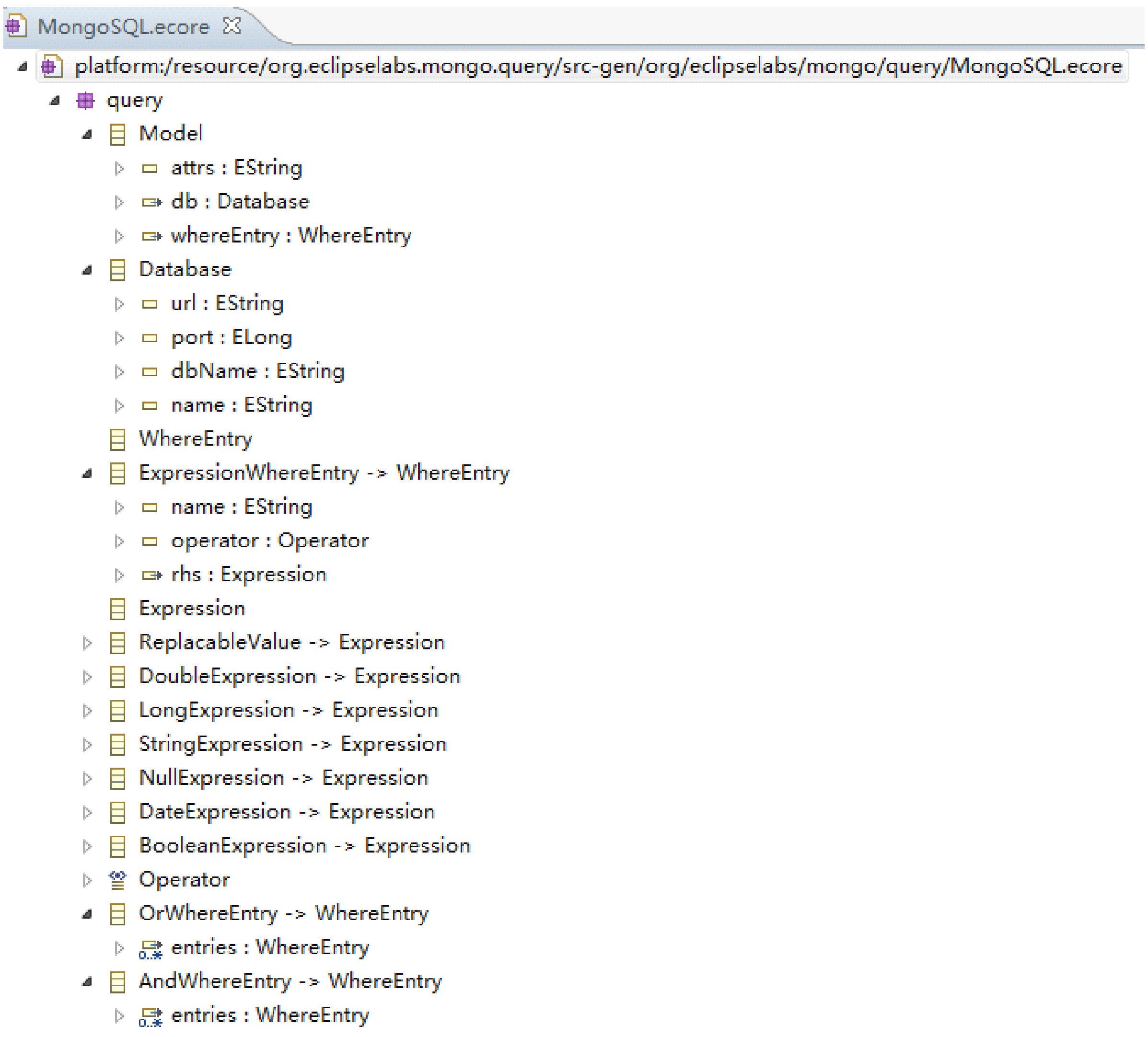
By default on a find operation, the entire object is returned. However we may also request that only certain fields be returned. This is somewhat analogous to the list of column specifiers in a SQL SELECT statement (projection) Retrieving a Subset of Fields.

```
db.things.find({ x:"john"},{ z:1});// select z from things where x="john"
```

So we would like to use SQL to be MongoEMF's query string, then user can build SQL to query data from MongoDB. An example use-case is as follows: A client constructs a query string(such as SELECT * FROM mongo://localhost/tigergui/users WHERE name="Test User7") included in a URI. The URI is sent to a server in a HTTP GET call. The server extracts the query string from the URI and re-builds the EMF query model. The query model is passed to the query engine and a DBObject is returned containing a MongoDB specific query.

1.2 EMF Query Model Class

EMF query model will be generated by XText technology. User can build query model instance, transfer it into MongoDB DBObject, then use this DBObject to query MongoDB. Simple query model demo ecore looks like this:



```
MongoSQL.ecore
platform:/resource/org.eclipselabs.mongo.query/src-gen/org/eclipselabs/mongo/query/MongoSQL.ecore
  query
    Model
      attrs : EString
      db : Database
      whereEntry : WhereEntry
    Database
      url : EString
      port : ELong
      dbName : EString
      name : EString
    WhereEntry
    ExpressionWhereEntry -> WhereEntry
      name : EString
      operator : Operator
      rhs : Expression
    Expression
    ReplacableValue -> Expression
    DoubleExpression -> Expression
    LongExpression -> Expression
    StringExpression -> Expression
    NullExpression -> Expression
    DateExpression -> Expression
    BooleanExpression -> Expression
    Operator
    OrWhereEntry -> WhereEntry
      entries : WhereEntry
    AndWhereEntry -> WhereEntry
      entries : WhereEntry
```

1.3 De-Serializer

This module is generated by XText and special grammar, can be used to transform SQL to EMF query model.

```
    //Transfer query language => EMF query model
    new org.eclipse.emf.mwe.utils.StandaloneSetup().setPlatformUri("../");
    Injector injector=newMongoSQLStandaloneSetup().createInjectorAndDoEMFRegistration();
    XtextResourceSet resourceSet= injector.getInstance(XtextResourceSet.class);
     resourceSet.addLoadOption(XtextResource.OPTION_RESOLVE_ALL,Boolean.TRUE);
    Resource resource= resourceSet.createResource(URI.createURI("dummy:/example.mongosql"));
    InputStreamin=newByteArrayInputStream("SELECT * FROM mongo://localhost:27017/tigergui/users WHERE name='Test
User7' AND age>2".getBytes());
    try{
            resource.load(in, resourceSet.getLoadOptions());
    }catch(IOException e){
            e.printStackTrace();
    }
```

```
ModelImpl model=(ModelImpl) resource.getContents().get(0);
```

The result model instance is an EMF query model class object, we can use this model to query MongoDB.

1.4 Serializer

This module will be used to transfer EMF query model into SQL based human readable query string.

```
Model dt= model;
 dt.setAttrs("*");
 resource.getContents().set(0, dt);
HashMap saveOptions=newHashMap();
 saveOptions.put(XtextResource.OPTION_FORMAT,Boolean.TRUE);
try{
     resource.save(newFileOutputStream("sample.mongosql"), saveOptions);
}catch(FileNotFoundException e){
     e.printStackTrace();
}catch(IOException e){
     e.printStackTrace();
}
```

This will result SQL string SELECT * FROM mongo:// localhost / tigergui / users WHERE name = 'Test User7' AND age > 20 OR sex != 'female'

2. Transfer Module

This module is used to transfer EMF query model class to DBObject which can be used to query MongoDB. Sample code is just like this:

```
//Transfer EMF query model => DBObject
ModelImpl model=(ModelImpl) resource.getContents().get(0);
Mongo m=null;
try{
     m=newMongo( model.getDb().getUrl(), model.getDb().getPort());
}catch(UnknownHostException e){
     e.printStackTrace();
}catch(MongoException e){
     e.printStackTrace();
}
 DB db= m.getDB(model.getDb().getDbName());
DBCollection coll= db.getCollection(model.getDb().getName());
 DB db= m.getDB(model.getDb().getDbName());
DBCollection coll= db.getCollection(model.getDb().getName());
BasicDBObject query=(BasicDBObject) tansferModule(model);
```

Method DBObject tansferModule(ModelImpl? model) is like this:

```
privateDBObject tansferModule(ModelImpl model){
BasicDBObject query=newBasicDBObject();
WhereEntry rootEntry=model.getWhereEntry();
if(rootEntryinstanceofExpressionWhereEntry)
        transferExpressionWhereEntry(rootEntry, query);
elseif(rootEntryinstanceofAndWhereEntry)
        transferAndWhereEntry(rootEntry, query);
return query;
}

privatevoid transferAndWhereEntry(WhereEntry entry,DBObject query){
AndWhereEntry root=(AndWhereEntry) entry;
EList<WhereEntry> list= root.getEntries();
for(WhereEntry e: list)
      if(einstanceofExpressionWhereEntry)
              transferExpressionWhereEntry(e, query);
}

privatevoid transferExpressionWhereEntry(WhereEntry entry,DBObject query){
ExpressionWhereEntry expression=(ExpressionWhereEntry) entry;
if(expression.getOperator()==Operator.EQUAL)
        query.put(expression.getName(), getValue(expression.getRhs()));
elseif(expression.getOperator()==Operator.GREATER_THEN)
        query.put(expression.getName(),newBasicDBObject("$gt",
Integer.parseInt(getValue(expression.getRhs()).toString())));
}

privateObject getValue(Expression expr){
if(exprinstanceofStringExpression)
      return((StringExpression) expr).getValue();
elseif(exprinstanceofLongExpression)
      return((LongExpression) expr).getValue();
returnnull;
}
```

Support query of "Retrieving a Subset of Fields", we can get a DBObject instance represent target field list:

```
    privateDBObject getColumnsFromModel(ModelImpl model){
    DBObject columns=newBasicDBObject();
    if(!model.getAttrs().equals("*")){
          String[] list= model.getAttrs().split(",");
          for(int i=0; i<list.length; i++){
                String column= list[i].trim();
                if(!column.isEmpty())
                        columns.put(column, i+1);
```

```
        }
    }
    return columns;
    }
```

Then, we can use query DBObject and columns DBObject to query MongoDB:

```
    DBObject query= tansferModule(model);
    DBObject columns= getColumnsFromModel(model);
    DBCursor cur= coll.find(query, columns);
    while(cur.hasNext()){
            System.out.println(cur.next());
    }
```

Demo & Job finished so far

I have finished a demo query engine for MongoEMF which is using SQL as its human readable query string, just like: SELECT id,name,age FROM mongo://localhost/tigergui/users WHERE name='Test User7' AND age>20 OR sex!="female"

Host the whole project - MongoEMF-query-engine in Github here.

MongoEMF user can use SQL to query MongoDB, suppose we have a table with following records:

```
    > db.users.find();
    {"_id":ObjectId("4f65967bd7fdfafd4d45cece"),"name":"Test User1","age":1}
    {"_id":ObjectId("4f65967bd7fdfafd4d45cecf"),"name":"Test User2","age":2}
    {"_id":ObjectId("4f65967bd7fdfafd4d45ced0"),"name":"Test User3","age":3}
    {"_id":ObjectId("4f65967bd7fdfafd4d45ced1"),"name":"Test User4","age":4}
    {"_id":ObjectId("4f65967bd7fdfafd4d45ced2"),"name":"Test User5","age":5}
    {"_id":ObjectId("4f65967bd7fdfafd4d45ced3"),"name":"Test User6","age":6}
    {"_id":ObjectId("4f65967bd7fdfafd4d45ced4"),"name":"Test User7","age":7}
    {"_id":ObjectId("4f65967bd7fdfafd4d45ced5"),"name":"Test User8","age":8}
    {"_id":ObjectId("4f65967bd7fdfafd4d45ced6"),"name":"Test User9","age":9}
    {"_id":ObjectId("4f65967bd7fdfafd4d45ced7"),"name":"Test User10","age":10}
```

Query MongoDB by following code:

```
    //Transfer query language => EMF query model
    new org.eclipse.emf.mwe.utils.StandaloneSetup().setPlatformUri("../");
    Injector injector=newMongoSQLStandaloneSetup().createInjectorAndDoEMFRegistration();
    XtextResourceSet resourceSet= injector.getInstance(XtextResourceSet.class);
     resourceSet.addLoadOption(XtextResource.OPTION_RESOLVE_ALL,Boolean.TRUE);
    Resource resource= resourceSet.createResource(URI.createURI("dummy:/example.mongosql"));
    InputStreamin=newByteArrayInputStream("SELECT * FROM mongo://localhost:27017/tigergui/users WHERE name='Test
User7' AND age>2".getBytes());
    try{
            resource.load(in, resourceSet.getLoadOptions());
    }catch(IOException e){
        e.printStackTrace();
    }
    ModelImpl model=(ModelImpl) resource.getContents().get(0);
    //Transfer EMF query model => DBObject
    ....code snappetinTransferModule
    //Query MongoDB by DBObject
    DBCursor cur= coll.find(query, columns);
    while(cur.hasNext()){
    System.out.println(cur.next());
    }
```

Then the query result is:

```
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced4"},"name":"Test User7","age":7}
```

And if we change the SQL to "SELECT * FROM mongo://localhost:27017/tigergui/users WHERE age>2", the search result will be:

```
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced0"},"name":"Test User3","age":3}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced1"},"name":"Test User4","age":4}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced2"},"name":"Test User5","age":5}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced3"},"name":"Test User6","age":6}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced4"},"name":"Test User7","age":7}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced5"},"name":"Test User8","age":8}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced6"},"name":"Test User9","age":9}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced7"},"name":"Test User10","age":10}
```

And if we change the SQL to "SELECT name FROM mongo://localhost:27017/tigergui/users WHERE age>2", the search result will be:

```
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced0"},"name":"Test User3"}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced1"},"name":"Test User4"}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced2"},"name":"Test User5"}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced3"},"name":"Test User6"}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced4"},"name":"Test User7"}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced5"},"name":"Test User8"}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced6"},"name":"Test User9"}
{"_id":{"$oid":"4f65967bd7fdfafd4d45ced7"},"name":"Test User10"}
```

Background

Background skills

I have downloaded MongoEMF months ago, learn how it works, what kind of search engine it need. And gathered enough information for this project, grasp all the necessary skills(score: 1-10, 10 means perfect):

- Eclipse plugin development: 9
- EMF: 8

- XText: 7
- MongoDB: 8

My time plan

Before June 30th, i can spend 4 hours on this project in working days, full time in weekend, it means about 35 hours per-week. My summer holiday will start at July 1st, then i can work for this search engine full time, at least 40 hours per-week (Not consider weekends).

Deliverables

1. Souce code (be merged into MongoEMF git repository)

In my plan, it will have 5 OSGi bundles:

- org.eclipselabs.mongo.query, contain XText query grammar relative generation code, include EMF model design for basic SQL etc. This bundle can be used to contribute back to EMF Query 2 project.
- org.eclipselabs.mongo.query.mongodb, contain MongoDB specific extensions to the query grammar, advanced SQL grammar with MongoDB features, used only by MongoEMF project.
- org.eclipselabs.mongo.query.transform, contain transform module relative source code.
- org.eclipselabs.mongo.query.main, supply new search engine's operation interface API to MongoEMF users/
- org.eclipselabs.mongo.query.tests, unit tests package

2. Extensive unit tests code

Unit test code will be packaged into bundle org.eclipselabs.mongo.query.tests. Developers can use these test cases to test the new search engine.

3. End-user documentation

Documentation will be contributed to the UserGuide on the MongoEMF wiki.

Schedule

1. Before March 26: Prepare basic skills, gather demands and finish a Xtext and EMF based prototype query engine.

2. March 27 - April 6: Prepare project proposal

3. April 7 - May 20: Community bounding period

4. May 21 - May 30: Finish EMF query model design

5. June 1 - June 30: XText grammar design and implement, should cover all the MongoDB's build in functions support.

6. July 1 - July 15: Finish transfer module which can transfer EMF query model to DBObject of

MongoDB's java driver.

7. July 16 - July 31: Write unit tests to test the new engine, fix bugs.

8. August 1 - August 13: Improve documents, write user guide, development guide etc.

9. After August 13: 'pencils down' date for GSoC project, prepare for final evaluation.

Expectations

By this project, i want to learn how to work with open source community well. Although GSoC 2012 not really start yet, but i have learn much from open source community, from my potential mentor - Bryan Hunt. I hope i can be a MongoEMF committer after GSoC, continue to provide bug fixes and enhancements after the project is complete, and become an Eclipse community committer once MongoEMF become an advanced Eclipse project in the future.

Contact Information

My name is Tiger Gui, come from Beijing, China. My major is Computer Technology, so programming and database is part of my coursework, and Java is my favorite language. Now i am familiar with Java, Eclipse plugin development, OSGi and Eclipse opensource technology such as EMF. I am cccustomed to host my web application in equinox + osgi Jetty bundle, MongoDB is my favorite NoSQL solution which i am already familiar with.

So once i saw this project idea in Eclipse GSoC 2012 ideas page, i decided to communicate with Bryan Hunt and discuss this project in mail list even Twitter. Now i have all the basic skills for this project, and finished prototype query grammar, i think we can finish a fantastic project together this summer.

My email and skype account is tigergui1990@gmail.com, and twitter account is @tiger_gui :-)

Links

- 1.https://github.com/BryanHunt/mongo-emf
- 2.https://github.com/tigergui1990/MongoEMF-query-engine
- 3.http://www.mongodb.org/display/DOCS/Mongo+Query+Language
- 4.http://www.mongodb.org/display/DOCS/Retrieving+a+Subset+of+Fields